Amendments to the drawings:

Applicant has provided drawings sheets labeled "Replacement Sheet" for the following figures:

Figure 2, which changes the input to sequence controller 130 to "Power-on 131"

Figure 3a, which changes the label for "High Speed Bus" to "202"

Figure 5, which adds "Y" to 504, and "Y" and "N" to 516 and 518.

Amendments to the specification:

Please replace the paragraph starting on page 6 line 18 with the following paragraph:

--A wireless receiver 100 comprises a ROM 116 (Read Only Memory), a ROM Controller 114 coupled to the ROM 116 and also to the ~~low-speed data~~ low speed bus 123 having an address 122 and data 124, a bridge 110 coupling the low speed ~~data~~ bus 123 to a ~~high-speed data~~ high speed bus 119 having an address 118 and data 120, a DMA (Direct Memory Access) controller 126 coupled to the high speed ~~data~~ bus 119, along with static memory 104, dynamic memory 106, and a Wireless Front End (WFE) 108 coupled to the low speed ~~data~~ bus 123. The wireless front end 108 is coupled to a remote wireless host 128 which contains executable operating system code for use by the wireless receiver 100. Upon power-up, the ROM controller 114 reads the three data values SRC (Source), DST (Destination), and LENGTH from the ROM 116 and translates these three values to the address of the SRC, DST, and LENGTH registers of the DMA controller 126. The ROM Controller 114 resides on the low-speed bus 123, and the DMA controller 126 resides on the high speed bus 119. The Bridge 110 automatically couples and translates addresses and data from the low speed bus 123 to

the high speed bus 119 to enable this transfer of data. Once the DMA controller 126 has these three values, it automatically begins a Direct Memory Access sequence, whereby it transfers a LENGTH number of bytes of data specified by the contents of the SRC address to the DST address. If the SRC address points to the ROM contents as accessed by the ROM controller, and the DST points to the memory such as SRAM 104, the data is automatically copied from ROM 116 to SRAM 104. The CPU 102 remains in an inactive reset state during this transfer. Once the CPU 102 is taken out of reset, it begins executing the code that was earlier copied from ROM 116 into memory 104, which also contains the bootup sequence sufficient to begin the downloading of code from the ~~wireless host 126~~ wireless host 128. The CPU 102 requests the balance of code to be downloaded from the wireless front end 108 which is coupled to a wireless host 128, and it is copied into DRAM 106. When the download is completed, the CPU has the operating system image required for full operation.--

Please replace the paragraph starting on page 8 line 23 with the following paragraph:

--Figure 2 shows a wireless system 100, which has a high speed bus 119 comprising an address bus 118 and a data bus 120, as is known to one skilled in the art. There may also be a low speed bus 123 comprising an address bus 122 and a data bus 124. A bridge 110 couples the high speed bus 119 to the low speed bus 123, as is known to one skilled in the art of bridges. The busses are shown with separate address and data, as may be realized in one embodiment, although the busses could also be a single multiplexed address and data bus, such as PCI (www.pci.org), or SysAd bus of R7000 (www.pmc-sierra.com). When a device is controlling a bus through the issuance of read or write requests, it is referred to as a "bus master". Bridge 110 is bi-directional, such that bus masters may be located on the low speed bus 123 or the high speed bus 119 with contents 200 and 202, respectively, shown in figure 3a and 3b. On the low speed bus 123, devices which may act as bus masters are ROM controller 114, bridge 110, and wireless front end 108. On the high speed bus 119, devices which may act as bus masters are the CPU 102 and DMA controller 126. The bridge 110 may translate addresses and data from the high speed bus 119 to

the low speed bus 123, or it may preserve them, as one skilled in the prior art of big-endian and little-endian data translations, or data bus width adaptations is aware. In the present invention, bridge 110 requires no initialization, and performs the bridging bi-directionally upon power-up. A sequence controller 130 responds to a power-on signal 131, or any signal which indicates the boot sequence is to begin. The sequence controller 130 thereafter generates a ROM controller enable 132, which starts a first sequence of events, and generates a CPU_EN signal 134 at a later time.--